REMARKS

Claims 1-20 are currently pending in the application. By this amendment, claim 13 is canceled and claims 1, 6, 12, 14 and 20 are amended for the Examiner's consideration. The foregoing separate sheets marked as "Listing of Claims" shows all the claims in the application, with an indication of the current status of each.

The Examiner has rejected claims 1-2, 6-7, 11-15, 17 and 19-20 under 35 U.S.C. §102(a,e) as being anticipated by U.S. Patent No. 6,473,748 to Archer. The Examiner uses Archer primarily for the Common Object Request Broker Architecture (CORBA) described in summary form in the background section of Archer, not for the invention itself disclosed by Archer. CORBA is a standard which allows software modules to communicate with other software modules or other programs written in different programming languages or running on different platforms (col. 1, lines 21-27). Also, CORBA "was designed to allow intelligent components to discover each other and interoperate on an object bus" (col. 2, lines 46-48), but goes beyond interoperability to specify "an extensive set of bus-related services" (col. 2, lines 48-50). Interfaces between objects "are written in a neutral Interface Definition Language (IDL) that defines a component's boundaries" in such a fashion as to be "accessible across languages, tool, operating systems, and networks" (col. 2, lines 53-58). The structure of CORBA is such that a client module using another software module has no need to know where the other software module resides or on what operating system or how it is implemented (col. 2, line 64, to col. 3, line 2). The client module only needs to know the interface published by the other software module (col. 3, lines 5-7) in IDL. These "IDL-specified methods can be written in and invoked from any language that provides CORBA bindings" (col. 3, lines 9-10). CORBA provides an Object Request Broker (ORB), a software object bus that "allows software objects to discover each other at run time" and thereby "invoke each other's services" (col. 3, lines 26-35).

The above description goes beyond those passages cited by the Examiner, but provides a helpful basis for understanding how the Examiner has read the language of the claims. It is believed that by framing the discussion in this slightly broader view of CORBA it will clarify whether, and how, the claims need to be amended to distinguish the invention from the CORBA architecture. The CORBA architecture can be viewed as a way of making a variety of software modules, written in different languages and under different operating systems located at various places on an enterprise network, available to a client module. Of course, the fact that these software modules may be written in different languages and operating systems located at various places on a network is irrelevant to the present invention, although it is central to the design of CORBA. It would appear that any software module that publishes an IDL interface is available to any client, and that the client module and the other software module may "discover each other at run time" (col. 3, lines 33-34).

This suggests that these software modules with published IDL interfaces can serve as the claimed "platform of said service components" and that the client module (or an application program of such modules) can be viewed as the claimed "incomplete application" that uses these "reusable service components" to complete the application. In the CORBA scheme, under this interpretation, discovery of these "service components" at run time serves as "completion" of the application. The Examiner completes his analysis of claim 1 by citing reuse of the CORBA structure in "new or future enterprise computing needs" (col. 5, lines 45-46), to cover the maintenance features claimed.

However, this interpretation of the claims does not embody the invention as described in the specification. Indeed, this interpretation does not distinguish

CORBA from any of a large number of programming environments that employ libraries of callable/reusable code. It will be observed that the distinctive feature of the CORBA architecture – that it provides interoperability across programming languages and operating systems – has no relevance. The CORBA architecture still serves as a reference because the architecture could, in principles, be applied to a single application language (as with the present invention). But with interoperability removed as an issue, there is no difference between the CORBA architecture and any programming language which makes use of libraries of procedures or callable subroutines. Why isn't any such language a reference against claim 1 of the present invention? Surely a reference could be found (e.g. Archer) which applies such a language in an "enterprise environment" in order to develop and maintain an application responsive to "business process requirements" (e.g. Archer at col. 5, lines 10-17).

This points to an omission in the Examiner's analysis. There are several important limitations, already contained in the claims, missing from the Examiner's analysis. The first limitation is contained in the following claim language is:

> said reusable service components being built and deployed on an application server to facilitate completion and scaling of said incomplete application (emphasis supplied)

The Examiner's citation for this portion of the claim (col. 1, lines 27-30) says nothing about "scaling." As stated in response to the prior office action,

> It is an important aspect of the invention that these services be constructed to be easily scalable. Thus, they are single commands or narrowly focused tasks and are built directly on the server itself (page 4, lines 22-25; emphasis supplied).

The passage cited from the present application (page 4, lines 22-25) is instructive, and will now be quoted in full:

> **By building and deploying the most commonly used functions and services directly on the application server, the invention eliminates the need to scale an entire application**.

Where does the "need to scale an entire application" arise? It arises from the following circumstance, which explains both the need for – and the novelty of – the invention. Particular applications, over their lifetime, "grow in size and complexity to meet ever expanding requirements" (page 2, line 14). What happens as a result? In some circumstances, the user simply adapts to the application (page 2, line 18). These are the users that have no economic clout and have no choice but to adapt their business in order to be able to use the application, if the application is critical to their business. But "in the enterprise environment where the application is a support service to a business process, ..." (page 2, lines 19-20)

> "... it is the business process which drives the requirements and the application must be customized and customized again as requirements evolve. This repeated customization is often difficult, time consuming and costly. Consequently, there remains a need for a methodology for structuring applications in an enterprise environment **such that applications can be scaled and integrated, responsive to the evolving requirements of the business processes** supported by the enterprise environment, without having to break down the finalized code and customize the application in a time consuming and costly manner" (emphasis supplied; page 2, line 20, to page 3, line 2).

For these application there is a connection between the growth of the business and the need to scale the application, which drives repeated customization of the application as the business evolves. Thus, the novel idea of the invention is a methodology for dealing with the prior art problem, wherein "applications grow in size and complexity to meet ever-expanding requirements" (page 2, line 14) and consequently require repeated customizations. In the prior art, as viewed from the hindsight perspective of the present invention, these applications were completed prematurely (page 3, line 24, to page 4, line 1). The methodology has two primary aspects. First, the application development is structured to rely upon service components adapted to an so that problems of scaling are handled by

As with many inventions, there is no guarantee of a market. As will be understood by one skilled in the art, while the invention can usefully be applied within a single enterprise to a core application where the enterprise anticipates eventual growth and expansion requiring that their core application be "scaled up," the true profitability potential of the invention is for use by third party application providers whose applications are marketed to a variety of enterprises and serve as core applications for these enterprises. Under the prior art, as stated above, "it is the business process which drives the requirements and the application must be customized again and again as requirements evolve."

The present invention holds out the promise of making that customization process much more efficient, but for marketplace success of the invention it will be necessary for both application developers, on the one hand, and those in control of enterprise environments, on the other hand, to buy into the methodology of the invention. If that happens, there will be developed for enterprise platforms a library of "server bullets" that are designed for scalability, and an application developer will develop an "incomplete application" (page 4, line 2) that can be both a) completed in the first instance in a specified enterprise environment (page 4, lines 4-5) and b) maintained by repetition of the completion process as the business requirements of the enterprise change (page 4, lines 14-16).

One further omission from the Examiner's analysis should be noted in connection with the above description of how the invention would operate in the marketplace. The claim language uses the phrase "vertical enterprise area containing the specified enterprise environment" (emphasis supplied). This language is supported in the specification (page 3, lines 10-12; see also page 4, line 8) and is well understood (e.g. "vertical markets") by those skilled in the art. The citation used by the Examiner to address this limitation (Archer; col. 4, lines 8-19) discusses consistent implementation of rules for a domain (e.g. invoicing, sales, manufacturing) across business boundaries (e.g. a sales application may invoke employee rules). The cited reference says nothing about vertical enterprise area. This is significant because, in the contemplation of the invention and as would be understood by those skilled in the art, "vertical" refers up and down the supply chain to the customers and suppliers in a given "enterprise area." For example, an automobile manufacturer purchases supplies from a parts supplier and distributes automobiles to dealers. The parts suppliers, manufacturers, and dealers are all in a "vertical enterprise area." The developer of a core application in this "vertical enterprise area" will be mindful not only of "server bullets" in the enterprise environment of an automobile manufacturer but also "server bullets" in the enterprise environment of parts suppliers and dealers who, in modern automated commercial practice, will have to use the application. The invention may well improve current practice, where use of the application is simply forced upon parts suppliers and dealers by the automobile manufacturer. The reference cited by the Examiner does not address the vertical enterprise area aspect of the invention, and therefore Archer fails as a reference on this ground as well as the above stated ground having to do with scaling.

As will be evident from the pages of the specification cited above, the significant language in the claims concerning scaling and vertical enterprise area as described in the specification was not addressed by the Examiner in his analysis. In short, these important claim limitations have not been accounted for by the references cited by the Examiner. It is therefore believed that this ground of rejection put forward by the Examiner is not adequate under the anticipation standards of §102 and should be withdrawn. Such withdrawal is respectfully requested.

However, in order to advance the prosecution of this case, an amendment to the independent claims is proposed, in light of the foregoing discussion, which would highlight and further clarify the above described and claimed aspects of the invention.

The Examiner has rejected claims 3-5, 8-10, 16 and 18 under 35 U.S.C. §103(a) as being unpatentable over Archer in view of "Enterprise Java Beans" to Monson-Haefel. Since these claims depend from claims now believed to be in allowable form, this ground of rejection is also overcome.

The present invention is a new paradigm for enterprise software development. The idea is that this kind of environment, particularly with respect to other enterprise environments in a vertical enterprise area, will make it easy for application developers to develop applications which are "incomplete" in the sense that they are structured to avoid the conventional and prior art "customization" step (shown in Fig. 4A and described at page 9, lines 3-9) occasioned by growth of the enterprise and consequence scaling demands. Instead, an enterprise and those in its vertical enterprise area with whom it has automated commerce develop (or, perhaps, have provided to them by the application developer) "server bullet" services (i.e. "reusable service components", item 430 in Fig. 4B; page 9, lines 12-13) that are responsive to the scaling demands of changing business requirements of the enterprise. The application is not "completed" until after this platform of "server bullet" services has been prepared for the enterprise environment (page 9, lines 9-15). The application is then "completed" by a step that links to the "server bullet" services provided by a particular enterprise environment.

This methodology for application development has two main advantages. First, it avoids scalability issues by leaving to the enterprise environment itself – via "server bullets" – the task of providing the services that would need to be scaled. Of course, the precise services thus made available for a particular enterprise environment will not be known until after the enterprise environment has been prepared, which is why the application is left in an "incomplete" state (page 9, lines 20-21). Second, by virtue of this methodology, the "completion ... in lieu of customization" step (page 9, lines 13-15) is designed to come at the end and can be repeated easily, so that individual "incomplete" applications can not only be adapted to different enterprise environments but "re-adapted" to changes in a particular enterprise environment. See the discussion in connection with Fig. 4C (page 10, lines 1-11) for an explanation of how "completion" in accordance with the invention differs from "customization" under the prior art.

Thus the "re-usability" described in CORBA and in Archer does not track the re-usability claimed for "server bullet" components. Under the CORBA architecture software modules are "re-used", but this is the ordinary reuse characteristic of any computer software routine that is available to be called by a variety of other routines. And, of course, the "server bullets" of the present invention also have that aspect, although even in that aspect the present invention is distinguished because the meaning of "reusable" that is properly and distinctly descriptive of the invention relates to the reuse of a "server bullet" component in other enterprise environments and in other vertical pre-configuration bundles. The claims have been amended to further clarify this point, as described above with respect to the limitation that uses the phrase vertical enterprise area.

In summary, with respect to the Archer reference in particular and the more general prior art of programming techniques for reusability of program code, the present invention is not concerned with merely providing a library of callable functions that can be brought together in an enterprise environment as a basis for developing and maintaining an application. Instead, the present invention provides a novel approach – in essence, a paradigm shift – to the design and construction of enterprise environments in a vertical enterprise area, taking into account the need to facilitate scaling of enterprise applications in response to growth and changing business requirements.

This provides a two pronged marketplace dynamic attractive to both application developers and enterprise developers. Using the invention, application developers have a methodology for designing their applications to be easily "completed ... in lieu of customization" to a wide variety of enterprise environments, each "completion" being done "later" (page 9, line 14) after a suitable enterprise platform has been prepared with "server bullets. Similarly, developers of an enterprise environment can use the invention to provide a highly scalable platform that is both tailored to serve the particular needs of the enterprise and also able to accommodate easy "completion" of "incomplete" applications.

In view of the foregoing, it is requested that the application be reconsidered, that claims 1-20 be allowed, and that the application be passed to issue.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at 703-787-9400 (fax: 703-787-7557; email: clyde@wcc-ip.com) to discuss any other changes deemed necessary in a telephonic or personal interview.

If a further extension of time is required for this response to be considered as being timely filed, a conditional petition is hereby made for such extension of time. Please charge any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 50-2041.

Sincerely,

Clyde R Christofferson
Reg. No. 34,138

Whitham, Curtis, Christofferson & Cook, P.C.
11491 Sunset Hills Road, Suite 340
Reston, VA 20190
703-787-9400
703-787-7557 (fax)

Customer No. 30743